コンピュータセキュリティ 15－2
（２００１．１２．２１）

着信ポート番号のランダム化によるサーバー防衛

力武 健次　菊地 高広　永田 宏　濱井 龍明　浅見 徹

株式会社 KDDI 研究所 高信頼 IP ネットワーク技術プロジェクト
〒356-8502 埼玉県上福岡市大原 2-1-15 Tel: 0492-78-7303
email: kenji@kddilabs.jp

本稿では着信するポート番号をランダム化することで、インターネット上のサーバーへの攻撃の耐性をより高める方法について考察する。インターネットでは、着信ポート番号で利用するアプリケーションサービスを区別する。DoS（サービス拒否）攻撃の多くはこの特性を利用し、特定のポート番号に対してのみ接続を試みることで、攻撃の効率を高めている。仮に 1 つのアプリケーションサービスに対して複数の着信ポートを用意し、通信内容を分散させることができれば、無線通信でのスペクトラム拡散変調方式と同様に DoS 攻撃などの接続妨害行為に対してサーバーの対攻撃性を高めることができる。本稿ではこのモデルの適用可能性について検討する。

キーワード: インターネット、ネットワークセキュリティ、情報隠蔽

Defending Servers by Randomizing Listening Port Numbers

Kenji Rikitake, Takahiro Kikuchi, Hiroshi Nagata, Tatsuaki Hamai and Tohru Asami

High Quality Internet Project, KDDI R&D Laboratories Inc.
2-1-15 Ohara, Kamifukuoka City, Saitama 356-8502 JAPAN / phone: +81 492 78 7303
email: kenji@kddilabs.jp

In this paper, we study the feasibility of increasing resiliency against attacks to Internet servers by randomizing the listening port numbers. On Internet, each application service is identified with the listening port number. DoS (Denial-of-Service) attackers take the advantage of this characteristics, by focusing the destination port number to maximize the efficiency of the attacks. If an application service uses multiple listening ports and diverse the traffics to the ports, the server becomes more resilient against connection-interference activities such as DoS attacks, as in the case of spread-spectrum modulation on radio communication. We analyze the applicability of this port-randomizing model.

Keywords: Internet, Network security, Information hiding

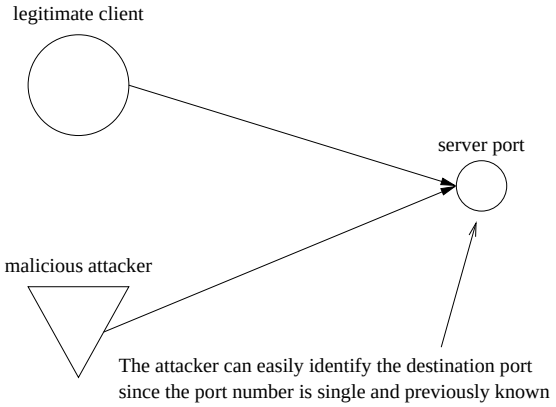

Fig. 1: Traditional single-port listening model

1 Introduction

Internet servers are usually designed to provide the services by listening to a specific *port*, waiting for incoming service requests, as in Figure 1. Attackers gain advantage from this model of Internet communication to focus to the targets and to make most of their resources for efficient attacks. For example, many known vulnerabilities are found in well-known services, such as SSH (Secure Shell) [20], RPC (Remote Procedure Call) [15], and Microsoft's NetBIOS service ports. Recent successful attacks such as *Nimda Worm* [1] specifically concentrate on attacking these known ports to discover the vulnerabilities, as listed in CERT/CC's Current Activity notes [2]. The notes also contains a list of vulnerable well-known port numbers.

The numbers of listening ports for well-known Internet services are globally publicized. This encourages DoS attackers to use the DDoS (Distributed Denial-of-Service) clusters of systems to practically paralyze production-level Internet services. While many proposals to defend systems from known DDoS attacks are published [4] [8], the Internet itself still does not have the mandatory mechanism to prevent address spoofing. Therefore, the server administrators still need to implement workaround methods to mitigate the risk of losing serviceability, such as increasing the processing power of the server clusters of the network bandwidth.

Recent deployment of Web-based computing environments also makes many devices such as routers and print servers rely on HTTP (HyperText Transfer Protocol) [17] for the configuration and management. These Web-aware devices open the HTTP port to public as default. Schneier warns that this proliferation of HTTP indicates means a single Internet-wide point-of-failure relying on a single port or service for different kind of service needs [3]. A good example that supports the Schneier's claim is that many routers were *abnormally* terminated operation due to the unexpected access from the Nimda and similar types of worms.

In this paper, we propose a model of using *randomized listening ports* for Internet servers to accept processing requests, to prevent DoS and similar types of attacks to a single port. Our model mitigates the interference caused by the attackers, and increase the tolerance and resiliency against the service-disruption attacks. We also discuss the implementation issues to effectively introduce this model for actual server operation.

2 Port Numbers and Internet Services

In this section, we describe the definition of the port numbers and the details of the usage.

Internet services assigned to a single IP (Internet Protocol) [10] address are uniquely identified by their destination *port numbers*, in either TCP (Transmission Control Protocol) [12] or UDP (User Datagram Protocol) [9]. The port numbers have a significant role to identify a communication channel and service provided over Internet.

Each TCP connection, a virtual bi-directional reliable data stream, is identified by two pairs of the IP addresses and the port numbers, one for the source and the other for destination. The source first attempts to connect to the destination and establish a connection.

In UDP data exchange, the source and destination pairs of the IP addresses and the port numbers are also used solely to identify the route and the usage of the packets. UDP has no notion of connection, since it does not establish a connection before exchanging data.

Each port number has a 16-bit length. The assignment of the port numbers is globally administered by IANA (Internet Assigned Numbers Authority) [19]. For example, HTTP is assigned to TCP port number 80. Since UNIX-derived oper-

Range	Name of The Ports
0-1023	Well-Known Ports
1024-49151	Registered Ports
49152-65535	Dynamic and/or Private Ports

Table 1: IANA Assigment Guideline for Port Numbers

ating systems do not allow non-privileged users to run a program listening to the ports of the number between 0 to 1023, those ports are commonly called *privileged ports*. The privileged ports are assigned to be used as the connect ports for the basic service protocols such as DNS (Domain Name System) [13] [14] and SMTP (Simple Mail Transfer Protocol) [18].

IANA periodically updates the list of the port assignment to the services [7]. IANA has the port number allocation plan as in Table 1. IANA calls the privileged ports as *Well-Known Ports*. IANA reserves the port numbers from 1024 to 49151 as *Registered Ports* and discourages users from using them for unregistered services. IANA also reserves the port number range from 49152 to 65535 as *Dynamic and/or Private Ports*.

Most of the port number space, however, is not strictly managed. Port numbers are only a tool for locating Internet services. If the client and server have a prior agreement, they can use a non-standard port number, since using a different port number does not affect the behavior of the transport protocols. For example, HTTP URL (Uniform Resource Locator) has a syntax to explicitly specify a non-standard port, such as `http://www.example.com:8080/`. In this case, the client attempts to connect to TCP port 8080 of `www.example.com`, instead of the standard port number 80.

The port number space has not been filled with assignment yet. As of November 13, 2001, IANA has officially assigned 3574 TCP and 3564 UDP ports from the port numbers between 0 and 49151 (49152 ports total), including 693 TCP and 691 UDP ports from the Well-Known port number range [7]. No port is assigned from the Private port number range either, from the 16384 ports. This indicates that an application program may use more than 60000 ports simultaneously at maximum excluding the Well-Known ports.

Using massive numbers of ports for one service, however, is not generally recommended since the port number space is shared by all services on a host. We recommend to use the Private port numbers for the model proposed in our paper, since it is rather experimental and non-standard.

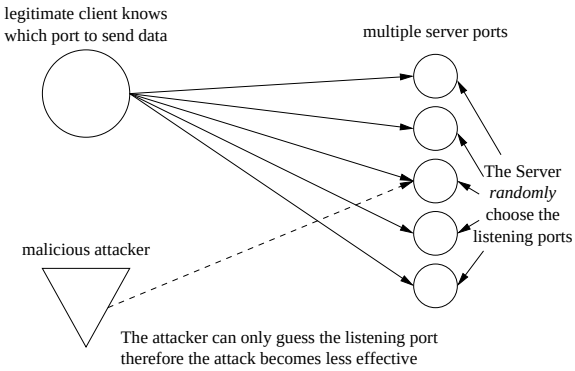

Fig. 2: Randomized multiple-port listening model

3 Randomization of Listening Ports

We propose a model of a server to *randomly* choose the listening port as shown in Figure 2, for reducing the interference caused by the attackers who send incomprehensible packets for the service disruption.

As described in Section 2, less than 10 percent of TCP and UDP port number space is assigned and and used for the current Internet services. Under this condition, using multiple ports for a service is feasible enough for the production-level application.

We assume the following conditions before discussing the implementation details:

- A server using multiple ports obtains no more bandwidth than a server using a single port. We can assume this unless a mechanism to allocate different network interfaces for each TCP/UDP port is available. In this paper we exclude this case since this is rather a traffic engineering issue that varies as the requirements of actual network environments change.
- The traffic characteristics of all the ports bound to a server is the same. The characteristics include, but not limited to, channel latency, packet loss rate, error rate, etc.
- A server can use the assigned ports solely for

itself. This means that a port is not shared with multiple servers for two or more services.
- A server simultaneously listens to a single port within the range of the allocated ports. While simultaneous listening to two or more ports is theoritically possible, this assumption helps simplifying the issues to be discussed.

We have a few choices to implement a server which simultaneously listening to the multiple ports as follows:

- *Using multiple UDP ports*. The traffic to a server is diffused into multiple UDP destination ports.
- *Using multiple TCP ports*. The traffic to a server is diffused into multiple TCP connections to many destination ports.
- *Using multiple UDP and TCP ports*. The traffic to a server is diffused into multiple TCP connections and UDP packets of many destination ports.

In this paper, we analyze the first model of the previous list, of using multiple UDP ports. Since UDP does not require establishment of connection and disconnection before and after using the ports, the implementation does not require the setup and clean-up procedures before and after allocating the ports. UDP does not resend the packets either; TCP retries sending lost packets, which may lead to unexpected delay of transferring packets.

The actual flow of communication using the proposed randomization of the port numbers using multiple UDP ports is as follows:

a) *Port-number sequence initialization:* the server and the client negotiates the PRN (Pseudo-Random Number) sequence generation method before beginning the communication session with a secure method. An example of the secure method is exchanging the information using an encrypted communication channel.
To avoid being DoS-attacked, the client and server may decide the initial port number without exchanging information, by using a shared secret and uniquely-defined identifiers such as time based on TAI (Temps Atomique International, or International Atomic Time) and the source and destination IP addresses or domain name strings.

b) *Data exchange:* the client send the data packet to the port number agreed in the negotiation phase mentioned in **a)**. The server sends the acknowledgement packet from the port of the same port number it receive the data packet to the client, to identify itself. Each packet contains a sequence number so that the server or client can calculate the correct port number using the PRN generator even if the packets are lost.

c) *Choosing next port number:* after each exchange described in **b)** is finished, the sequence number is incremented for the server and client, to recalculate a new port number for next exchange of packet. The **b)** and **c)** actions continue until the communication session ends.

d) *Re-initializing the port-number sequence:* for a connection-less communication, the initialization sequence of **a)** should be done when requested either by the client or by the server. For a connection-based communication, the sequence should be performed before each connection.
In either connection-less or connection-based case, the re-initialization should be done when the usage of the PRN sequence exceeds the lifetime. The lifetime should be determined by the cryptographic strength analysis of the PRN-generation algorithm. Practically, the lifetime should not be longer than an hour.

4 The Advantages and Disadvantages

The main advantage we gain by the randomization of listening ports is the reduction of the interference caused by the attackers. The attackers should send the traffics to *all* ports to which a server is *possibly* listening, for the full disruption of the service. In another words, the maximun gain of the S/N (Signal-to-Noise) ratio of the server is multiplied by the number of the allocated ports. For example, if 100 ports are randomly used by a server, the number of the packets the attacker should be able to send is 100 times of those than the case when only one port is used, to fully block the communication link. We expect this advantage from the theoritical analysis of channel-hopping

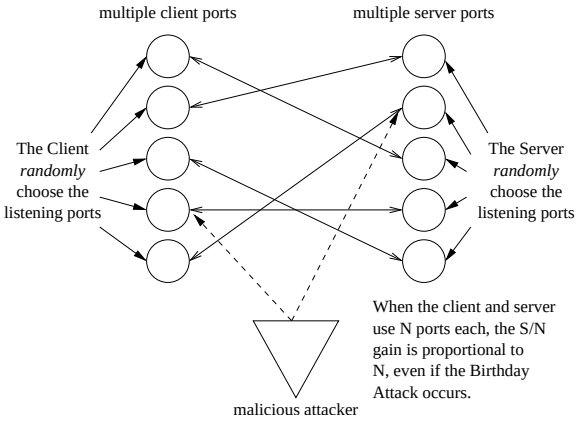

Fig. 3: Randomized multiple-port client-server model

spread-spectrum modulation method, commonly used on radio communication such as the IEEE 802.11b Radio LAN Standard.

When an attacker previously knows the port-number range and randomly choose the port to attack, however, the gain of the S/N ratio by choosing the multiple ports is reduced proportional to the square root of the number of ports simultaneously used by the server. This is a typical case of *Birthday Paradox Problem* [6]. This means that when a server allocates n ports, the S/N gain is proportional to $\sqrt{n}$. For example, the overall S/N gain by allocating 365 ports is about 23.

One of an alternative implementation to increase the S/N gain is to implement the port randomization also to the client using multiple ports as shown in Figure 3. This will increase the S/N gain to n when the client and server allocate n ports each, because the number of combinations of the port numbers is n^2.

We also observe that the randomization of listening ports have following disadvantages:

- Using the port randomization does not change the overall available bandwidth for a server or a network, so non-service-oriented DoS or DDoS, such as ICMP (Internet Control Message Protocol) [11] Echo packet flooding cannot be mitigated.
- Since multiple ports are used, packet filters for the server using the port randomization must allow wider range of incoming packet port numbers. This will affect the performance of the packet filters, and may allow unwanted packets coming in.
- The number of available ports is limited. Only 16384 ports are available as the IANA Private ports. This means the port randomization method cannot obtain the S/N gain more than 100, when using the Private ports only.

5 Implementation Issues

We need to consider implementation issues including, but not limited to, those listed in the following:

- The number of ports to which can be simultaneously listened is under restriction of the operating system. For example, FreeBSD 4.4-RELEASE [5] has the default limit number of 1024 (`FD_SETSIZE` of `<sys/types.h>`) for the file descriptors to listen to the ports, though this value can be changed.
- The PRN generator determines the strength against the port-specific attacks. The PRN must be cryptographically strong enough to avoid the sequence pattern to be revealed by a third party.
- To reduce the calculation time of PRN, a chain of one-way hash function should be considered to generate a sequence of the port numbers. This method has been widely used as A One-Time Password System [16] and proven to have enough practical cryptographic strength while minimizing the computational time of PRN generation.

6 Conclusion and Further Studies

The port randomization model presented is based on the fact that many Internet attacks are port-specific. We have shown that the proposed model can obtain the S/N gain proportional to the square root of the number of ports simultaneously used. While the model does not mitigate the brute-force attacks of consuming bandwidth, it contributes to reduce the server processing time by filtering out the interference packets before they are actually interpreted.

We have only proposed a model in this paper, so further studies by prototyping is needed to determine how to apply this model to the production systems, and to evaluate the traffic characteristics.

While the proposed model is effective in current IP version 4, the model can also be enhanced to use the address space. When the amount of available address numbers is vastly increased in the IP version 6 (IPv6), this model can be adopted to a combination space of the port and the address spaces, which will be exponentially increased from 16384 to 2^{62}, if 48 bits of an IPv6 address is used for the address randomization. This will significantly make this model effective to mitigate the risk of port-and-address-specific attacks. A further evaluation and prototyping on an IPv6 environment is required to prove the scalability of the proposed model.

Acknowledgements

We thank Sachiko Hirota, Yoshiji Sasano, Takao Hotta, Koji Nakao, and the other members of KDDI R&D Labotaroties Inc. for their support to our High Quality Internet Project.

We would also like to thank Dr. Shin-ichi Nakagawa of Communications Research Laboratory, and Dr. Mieko Kimura of Takeda Research Institute of Life Science, for their continuous support to our project activities.

References

[1] CERT/CC: *Nimda Worm*, CERT Advisory CA-2001-26 (last revised: September 25, 2001). http://www.cert.org/advisories/CA-2001-26.html

[2] CERT/CC: *CERT/CC Current Activity*, http://www.cert.org/current/current_activity.html

[3] B. Schneier: *Code Red Worm*, CRYPTO-GRAM, August 15, 2001, Counterpane Internet Security, http://www.counterpane.com/crypto-gram-0108.html

[4] T. Kikuchi, T. Asami, K. Rikitake, H. Nagata, T. Hamai: *Reverse-Path-Based Filtering against IP Address Spoofing*, IPSJ Computer Security Symposium CSS2001 Proceedings, IPSJ Symposium Series Vol. 2001, No. 15, pp. 191–196 (2001).

[5] The FreeBSD Project: *The FreeBSD Operating System, Version 4.4-RELEASE*, September 2001. http://www.freebsd.org/releases/4.4R/announce.html

[6] D. E. Knuth: *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison-Wesley (1973).

[7] Internet Assigned Numbers Authority: *IANA Port Numbers list*, http://www.iana.org/assignments/port-numbers

[8] V. Paxson: *An analysis of using reflectors for distributed denial-of-service attacks*, Computer Communication Review, Vol. 31, No. 3, pp. 38–47 (2001).

[9] J. Postel: *User Datagram Protocol*, RFC768 (also STD6) (1980).

[10] J. Postel: *Internet Protocol*, RFC791 (also STD5) (1981).

[11] J. Postel: *Internet Control Message Protocol*, RFC791 (also STD5) (1981).

[12] J. Postel: *Transmission Control Protocol*, RFC793 (also STD7) (1981).

[13] P. V. Mockapetris: *Domain names – concepts and facilities*, RFC1034 (also STD13) (1987).

[14] P. V. Mockapetris: *Domain names – implementation and specification*, RFC1035 (also STD13) (1987).

[15] R. Srinivasan: *RPC: Remote Procedure Call Protocol Specification Version 2*, RFC1831 (1995).

[16] N. Haller, C. Metz, P. Nesser, M. Straw: *A One-Time Password System*, RFC2289 (1998).

[17] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1*, RFC2616 (1999).

[18] J. Klensin, *Simple Mail Transfer Protocol*, RFC2821 (2001).

[19] J. Reynolds, R. Braden, S. Ginoza, L. Shiota: *Internet Official Protocol Standards*, RFC3000 (also STD1) (2001).

[20] D. J. Barrett, and R. E. Silverman: *SSH, The Secure Shell: The Definitive Guide*, O'Reilly & Associates, ISBN 0-596-00011-1 (2001).